

2- SOLUZIONE DI UN "PROBLEMA"

- ANALISI del problema, IDENTIFICAZIONE della soluzione
- DESCRIZIONE della soluzione in termini comprensibili all'esecutore
- INTERPRETAZIONE da parte dell'esecutore
- ATTUAZIONE della soluzione

CALCOLATORE \rightarrow caratterizzato da / linguaggio in grado di interpretare
/ istruzioni in grado di eseguire

2.1- PROBLEMI E ALGORITMI

Problema ELEMENTARE \rightarrow risolvibile con un'unica operazione -
(primitivo) "AZIONE ELEMENTARE"

Combinando più azioni elementari,
è possibile giungere a risultati attra. \Rightarrow SUDDIVISIONE
verso la risoluzione di un problema IN
iniziale non risolvibile con una azione SOTTOPROBLEMI
elementare

Soluzione EFFETTIVA PER UN ESECUTORE

l'esecutore è in grado di associare
le operazioni volte alla risoluzione
l'esecutore è in grado di
completare le operazioni in un tempo finito

PROBLEMA ELEMENTARE → Aspetto Descrittivo
AZIONE ELEMENTARE → Aspetto Esecutivo

Caratteristiche di un esecutore

- ~ linguaggio che interpreta
- ~ azioni che compie
- ~ insieme delle regole che associano costrutto linguistico ad una azione

FORMALE

Soluzioni
Formali



ALGORITMI

Descrizione
della
Soluzione di un problema

attraverso la
scomposizione
iterativa del
problema

- ~ il linguaggio è definito in modo formale → CARATTERIZZAZIONE SINTATTICA
- ~ azioni deterministiche = risultato costante
- ~ regole univocamente definite → CARATTERIZZAZIONE SEMANTICA

Nel caso del calcolatore:

ALGORITMI = PROGRAMMI
COSTRUITO = LINGUAGGIO DI PROGRAMMAZIONE

Sviluppo di un programma

- ANALISI problema; IDENTIFICAZIONE soluzione
- FORMALIZZAZIONE soluzione \rightarrow ALGORITMO \Rightarrow spesso eseguito da esseri umani, oppure
- PROGRAMMAZIONE \rightarrow scrittura al alto livello
- TRADUZIONE in linguaggio macchina \Rightarrow COMPILATORI con CASE INTERPRETI

any
language
interpreter
or
compiler
can
do
this
work

Problema: determinare il maggiore tra due numeri x, y
"P1"

Scomposizione problema

1) CALCOLARE DIFFERENZA TRA x e y

2) VALUTARE SE $E > DI 0$

\hookrightarrow se si, il maggiore e' x

\hookrightarrow se no, il maggiore e' y

} "PREDICATO"

STRUTTURE
CONDIZIONALI

Problema: determinare il maggiore tra n numeri
"P2"

1) TROVARE IL MAGGIORE TRA I PRIMI DUE (P_1)

2) FINCHÉ CI SONO NUMERI RIPETERE 3)

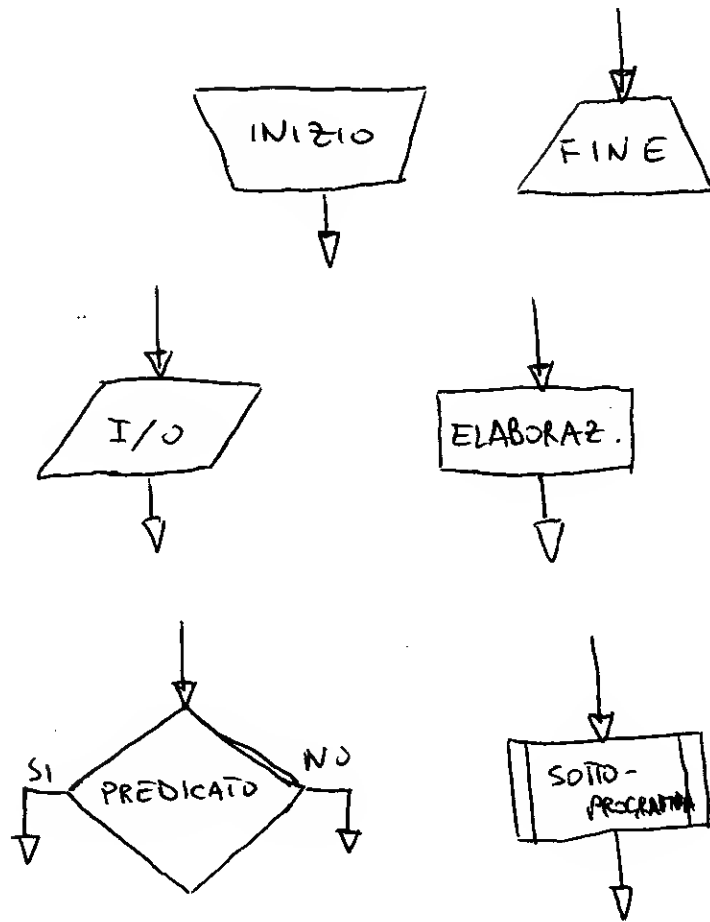
3) TROVARE IL MAGGIORE TRA IL PRECEDENTE
E IL NUOVO (P_1)

soluzione
concisa,
valida per un

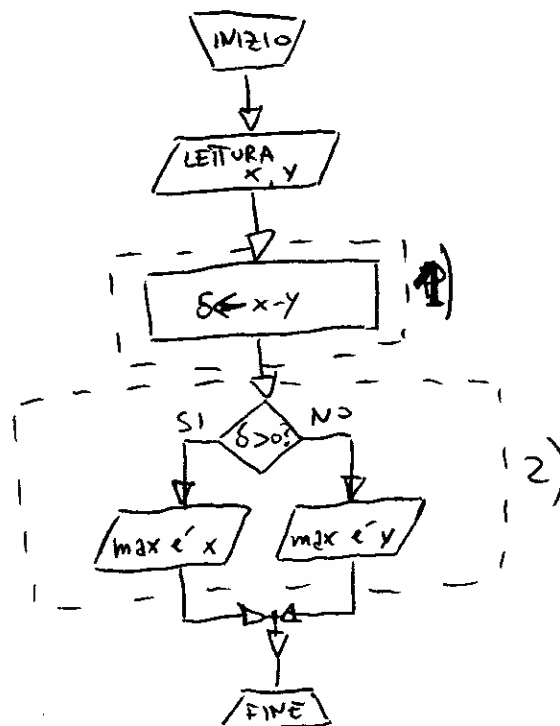
numero qualsiasi \rightarrow SOLUZIONE ITERATIVA, CICLO (loop)

RAPPRESENTAZIONE ALGORITMI CON DIAGRAMMI DI FLUSSO

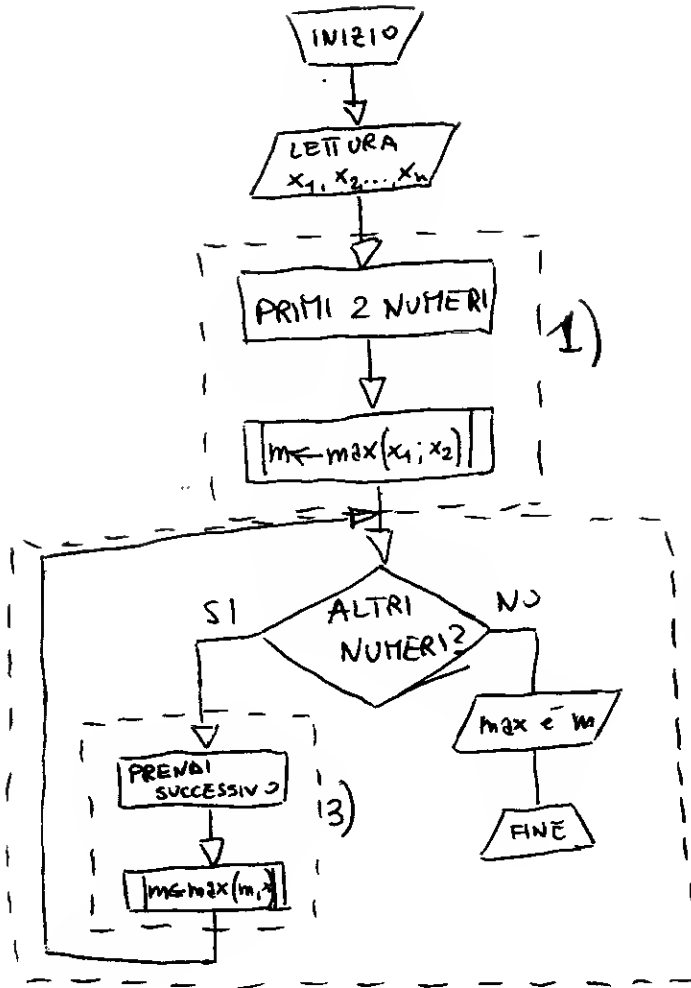
(FLOW CHART, DIAGRAMMI A BLOCCHI)



P1:



P2 :



$m \leftarrow \max(\dots)$

Corrisponde ad una completa esecuzione del primo programma

2.2 CODIFICA DEI DATI

Algoritmi \rightarrow AZIONI operate su DATI

↓
rappresentati in
un formato adatto

DATI \rightarrow Successione di simboli, scelti da un insieme finito
↓
detto ALFABETO

REGOLE DI COMPOSIZIONE : $\left\{ \begin{array}{l} 1234,5 \text{ è un numero} \\ 1,23,45 \text{ NON è un numero} \end{array} \right.$

CODICE \rightarrow relazione tra
successione ben formata di
simboli e il dato

nel caso dei
numeri, il codice è
la posizione $\rightarrow 123 \neq 321$

⇓
FORMALITÀ
dei
DATI

n SIMBOLI
k LUNGHEZZA
SUCCESSIONI $= n^k$

CODIFICA BINARIA

"0"; "1" \rightarrow utilizzata dai calcolatori \Rightarrow valore di tensione elettrica

BIT \rightarrow 1 elemento che assume
un valore binario

8 bit $\rightarrow 2^8 = 256$ valori rappresentabili

$2^6 \rightarrow 1 \text{ byte}$

$2^{10} \rightarrow 1 \text{ KByte}$

$2^{20} \rightarrow 1 \text{ MByte}$

$2^{30} \rightarrow 1 \text{ GByte}$

$2^{40} \rightarrow 1 \text{ TByte}$

BINARIO \rightarrow DECIMALE

$$101100_{(2)} = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 32 + 8 + 4 = 44$$

DECIMALE \rightarrow BINARIO

$$573_{(10)} = ?_{(2)}$$

$573 : 2 = 286$	r	1
$286 : 2 = 143$	r	0
$143 : 2 = 71$	r	1
$71 : 2 = 35$	r	1
$35 : 2 = 17$	r	1
$17 : 2 = 8$	r	1
$8 : 2 = 4$	r	0
$4 : 2 = 2$	r	0
$2 : 2 = 1$	r	0
$1 : 2 = 0$	r	1

$$573_{(10)} = 1000111101_{(2)}$$

SISTEMA ESADECIMALE₍₁₆₎

$\{0, 1, \dots, 9, A, B, C, D, E, F\}$

Ogni cifra esadecimale corrisponde a quattro cifre binarie

1010:1110:0001₍₂₎
↓ ↓ ↓ ↓
A, E, 1₍₁₆₎

CODIFICA BINARIA DI NUMERI INTERI

↓ Rappresentazione in COMPLEMENTO A DUE

↓
n bit numero x → $\text{compl}_2 = 2^n + x$

CODIFICA BINARIA DI NUMERI RAZIONALI

~ CIFRE PIÙ SIGNIFICATIVE

↓
723 456
↓
0,00136

~ NOTAZIONE SCIENTIFICA

$\pm m \cdot B^e$
↓ ↓
coefficiente base
(mantissa)

Nei calcolatori è sufficiente memorizzare segno, mantissa ed esponente.

IEEE → stabilisce dimensioni in bit

Institute of Electrical and Electronic Engineers

CODIFICA DI DATI NON NUMERICI

⑤

Per un insieme finito di n elementi,
trovare il numero j di bit che servono
per l'identificazione univoca degli elementi.

$$\textcircled{10} \quad m = 2^n \Rightarrow j = \log_2 n = \text{numero di bit necessari}$$

ES.: $n=13$

$$j = \log_2 13 = 3,7 = 4 \text{ bit} \Rightarrow \text{ARBITRARIETÀ della scelta dei codici}$$

||
V

CODICE $\rightarrow 7 \text{ bit} \rightarrow 128$
ASCII $(8 \text{ bit} \rightarrow 256)$
 $(16 \text{ bit} \rightarrow \text{Unicode})$

2.3 I PROGRAMMI

Programma \rightarrow sequenza di codice comprensibile
dal processore (linguaggio macchina)

SCRITTURA DI TALE CODICE
MOLTO COMPLICATA

↙
linguaggi ad alto
livello

← attraverso un
apposito software, il codice
scritto dal programmatore viene
trasformato in codice macchina

→ permettono di scrivere
il programma utilizzando
un'astrazione paragonabile
agli algoritmi

LINGUAGGI DI PROGRAMMAZIONE

sono caratterizzati da:

- ~ sintassi → insieme delle regole che specificano come creare istruzioni ben formate
- ~ semantica → specifica il significato, quindi l'azione, di ogni istruzione

VARIABILE → porzione di memoria nella quale

sono contenuti dati utili al programma

- TIPO della variabile: identifica le proprietà e le operazioni che possono essere svolte

- UTILIZZO della variabile:

- ~ dichiarazione
- ~ assegnamento

PROGRAMMA x MOLTIPLICAZIONE

```
main()  
{
```

Identificazione del programma

```
    unsigned int a, b;
```

```
    int w, z;
```

Dichiarazione variabili

```
    scanf("%d %d", &a, &b);
```

```
    z = 0;
```

```
    w = a;
```

```
    while (w > 0)
```

```
    { z = z + b;
```

```
      w = w - 1;
```

```
    }
```

```
    printf("%d", z);
```

```
}
```

Corpo del programma
(parte esecutiva)

Il programma viene eseguito dall'alto in basso così come è ~~in~~ scritto dal programmatore. Tuttavia esistono istruzioni "di controllo" che modificano questo ordine di esecuzione.

I DATI

- ~ unsigned int → numeri naturali (interi senza segno), spesso codificati su 32 bit
- ~ int → interi con segno, codificati su 32 bit
- ~ float → numeri in virgola mobile, su 32 bit, utilizzati per trattare numeri con decimali (7 cifre)
- ~ char → caratteri alfanumerici, 8 bit → adatti per l'ASCII
- ~ boolean → dati logici: vero/falso (true/false)

VARIABILI STRUTTURATE

vettori (array)
matrici (array multidimensionale)

~ ARRAY

~~int f[100];~~

```
main()
{
    int f[100];
    int w, z;

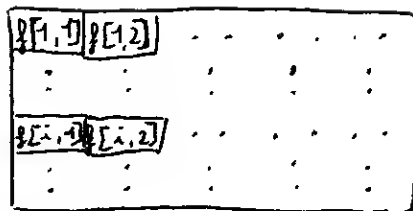
    w = 0;
    z = 0;
    while (w <= 99)
    {
        z = z + f[w];
        w = w + 1;
    }
    printf("%d", z);
}
```

vettove
num. di variabili
contenute

⇒ SOMMA DEGLI
ELEMENTI DI UN
VETTORE

~ MATRICI

$f[3,5]$



Nei vettori e nelle matrici, il tipo della variabile deve essere sempre uguale

DATI DEFINITI DALL'UTENTE

(USER-DEFINED, CUSTOM)

Una stessa variabile include più componenti (campi), ognuno con il proprio tipo

```
struct prodotto
```

```
{ char nome[dimensione];  
  int fatturato; };
```

```
...
```

```
main()
```

```
{ struct prodotto p;
```

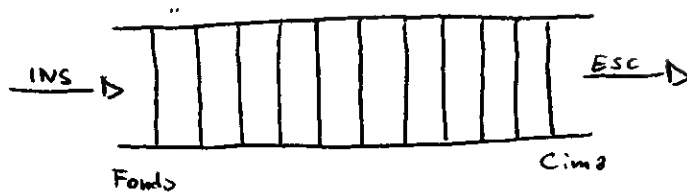
```
...
```

```
p.fatturato = ...; → assegna un valore al fatturato  
del prodotto p
```

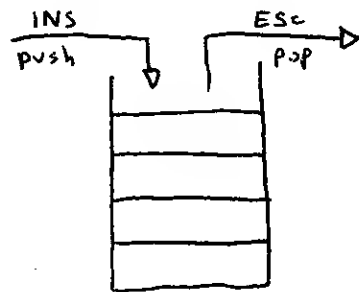
CODA delle variabili:
(queue)

↙ FIFO
↘ LIFO

~FIFO - first in, first out



~LIFO - last in, first out (stack = pila)



ISTRUZIONI

- 1) Istruzioni di ingresso-uscita
- 2) Istruzioni aritmetico-logiche
- 3) Istruzioni di controllo

1) Ingresso/Uscita (Input/Output)

Modalità standardizzate per l'acquisizione di dati e per la presentazione di dati.

- printf \rightarrow uscita
- scanf \rightarrow ingresso

2) Aritmetico-logiche

~ ASSEGNAZIONE (-)

$x = y$ \rightarrow Posiziona in x il contenuto di y
 $[x \leftarrow y]$ $b1 = h > j$

~ UGUAGLIANZA (==)

$(j == k) \rightarrow$ restituisce un valore true o false

~ FUNZIONI MATEMATICHE COMPLESSE

logaritmi

esponenziali

radici

⋮

~ OPERAZIONI LOGICHE

OR $\rightarrow ||$

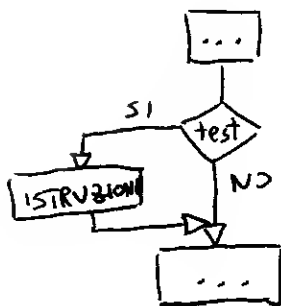
AND $\rightarrow \&\&$

3) Controllo

Le istruzioni di controllo sono quelle istruzioni che permettono di modificare il flusso di istruzioni, "saltando" ad istruzioni in un punto qualsiasi.

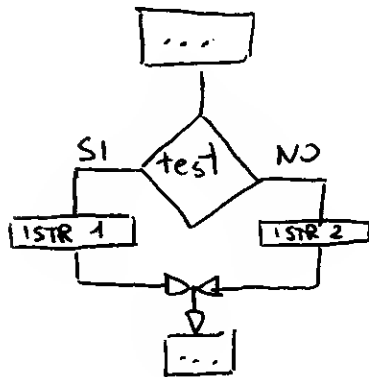
↖ SALTO INCONDIZIONATO
SALTO CONDIZIONATO

SALTI CONDIZIONATI



```

{ if (test)
  { /* istruzioni */ }
}
  
```



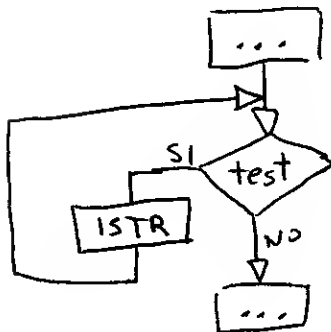
```

if (test)
{ /* istr 1 */ }
else
{ /* istr 2 */ }
  
```

STRUTTURE ITERATIVE (Loop)

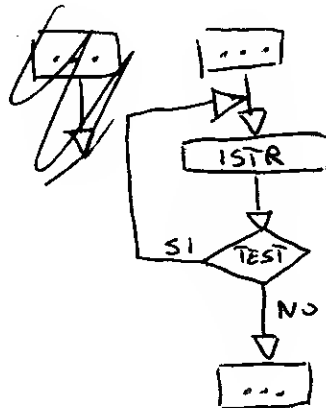
- ~ ciclo condizione iniziale
- ~ ciclo condizione finale

- CONDIZIONE INIZIALE



while (test)
{ /* istruzioni */ }

- CONDIZIONE FINALE



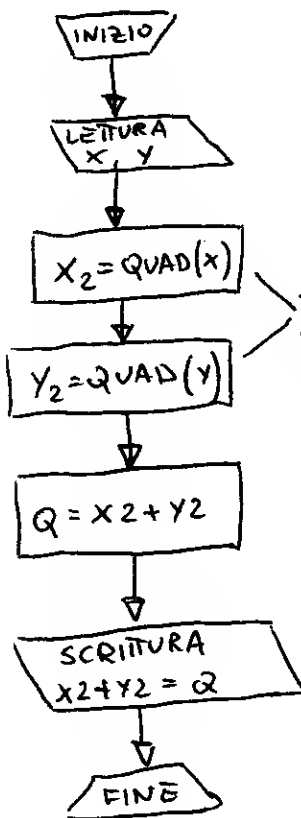
do
{ /* istruzioni */ }
while (test)

↓
in questo modo,
ISTR è eseguito
almeno una volta

STRUTTURAZIONE DEI PROGRAMMI IN SOTTOPROGRAMMI

Nella scrittura di un programma, è conveniente per la chiarezza e la leggibilità del codice utilizzare una struttura a sottoprogrammi, ovvero non ripetere più volte la scrittura di una data funzione più volte, ma predisporla un'unica volta per il funzionamento in più parti del programma e lavorando con valori che possono essere diversi di volta in volta.

PROGRAMMA PER IL CALCOLO DI $x^2 + y^2$



Si riferiscono ad una stessa funzione quad, scritta 1 volta ed eseguita 2 volte

```
int quad (int a)
{
    int w, z;
    w = a; z = 0;
    while (w > 0)
    {
        z = z + a;
        w = w - 1;
    }
    return (z);
}
```

```
main()
{
    int x, y, x2, y2, q;
    scanf("%d %d", &x, &y);
    x2 = quad(x);
    y2 = quad(y);
    q = x2 + y2;
    printf("%d", q);
}
```

11/11/11

3 ARCHITETTURA DI UN SISTEMA INFORMATICO - HARDWARE

Principali funzioni di un calcolatore

1. Elaborazioni dati
2. Memorizzazione dati
3. Trasferimento dati
4. Controllo

- 1)
- ~ FLESSIBILITÀ DEL CALCOLO → base non specializzata ed adatta a più applicazioni
 - ~ MODULARITÀ → ogni componente svolge una specifica funzione
 - ~ SCALABILITÀ → ogni componente può essere sostituito con un altro equivalente ma con prestazioni superiori
 - ~ STANDARDIZZAZIONE → componenti standard e quindi facilmente reperibili
 - ~ ABBATTIMENTO COSTI → produzione su larga scala
 - ~ SEMPLICITÀ INSTALLAZIONE
 - ~ DISPONIBILITÀ DI APPLICAZIONI a basso costo

2)

MEMORIZZAZIONE

↑ brevi periodi (elaborazioni intermedie)
tempi non definiti (sostituzione archivi cartacei)

3)

TRASFERIMENTO

↑ Comunicazione con l'esterno → periferiche (disp. di I/O)
su grandi distanze → trasmissioni dati (cbl. in rete)

4)

CONTROLLO

Coordinamento delle operazioni interne
e delle risorse del calcolatore

3.1 ARCHITETTURA DI FUNZIONAMENTO

Macchina di Von Neumann

- Interazione con l'esterno \Rightarrow dispositivi di I/O
- Unità di elaborazione centrale (CPU)
 - \hookrightarrow controllo e coordinamento operazioni
- Memoria \rightarrow dati elaborati e talvolta dati dell'I/O
 - \hookrightarrow celle adiacenti, identificate dall'indirizzo

COLLEGAMENTO A BUS

Bus \rightarrow linea che collega allo stesso
momento CPU con tutte le
unità

La CPU coordina e gestisce il
controllo del bus \Rightarrow ordine nell'assegnare
spazio alle periferiche

\Downarrow
non avvengono
collisioni di
dati

BUS

- SEMPLICITÀ → unica linea qualunque sia il numero di dispositivi → produzione economica
- ESTENDIBILITÀ → l'aggiunta di dispositivi è attuabile in poco tempo e senza modifiche al precedente hardware
- STANDARD → dispositivi di più produttori possono interagire
- LENTEZZA → incapacità di parallelizzazione delle operazioni
- CAPACITÀ LIMITATA → limite alla capacità di trasferimento dei dati
- SOVRACCARICO CPU → la CPU deve gestire tutte le operazioni di trasferimento

BUS

- bus dati → trasferimento dati CPU ↔ memoria / I/O
- bus indirizzi → identificazione celle di memoria
- bus controllo → selezione periferica, direzione (lettura o scrittura)

SCHEDA MADRE

→ alloggia CPU, BUS e principali dispositivi di I/O

ESECUTORE

Esecuzione di un programma:

- dati e istruzioni sono presenti in una memoria lettura/scrittura
- i contenuti della memoria sono indirizzati secondo la loro posizione
- le istruzioni sono eseguite sequenzialmente

↓
CPU => linguaggio macchina
(assembler)

codice operativo

CODIFICA ASSEMBLER	SOMMA	OPERANDO 1	OPERANDO 2
ES. DI CODIFICA IN LINGUAGGIO MACCHINA	10000011	001	010

↓
Codifica
binaria di una
istruzione

↓
POSIZIONI
DI
MEMORIA
(INDIRIZZI)

DIVERSE CPU

↓
ARCHITETTURA
diversa

→ Spesso ~~compatibili~~
compatibili

↓
ad esempio
INTEL e AMD
← stesso
linguaggio mac-
china

Esecuzione di un programma

↳ esecuzione ciclica della CPU di

- 1) lettura (FETCH) → acquisizione di un'istruzione
- 2) decodifica (DECODE) → identificazione dell'istruzione
- 3) esecuzione (EXECUTE) → effettuazione delle operazioni legate all'istruzione

✓
compresa lettura dei dati dalla memoria (operandi)

CPU

- ALU → unità aritmetico logica
⇒ operazioni matematiche
" logiche
- Registri → celle di memoria interne
- Unità di controllo → controllo delle operazioni e del BUS

REGISTRI

↳ memorizzazione operandi, esiti operazioni

- ~ PC (Program Counter) → indica la prossima istruzione da eseguire selezionandone la cella di memoria
- ~ IR (Instruction Register) → contiene una copia dell'istruzione da eseguire
- ~ MAR (Memory Address Register) → contiene l'indirizzo di memoria ove scrivere o leggere un dato → BUS INDIRIZZI
- ~ MDR (Memory Data Register) → dato da scrivere o già letto in memoria → BUS DATI
- ~ PSW (Processor Status Word) → informazione esito di un'operazione (RISULTATO)
- ~ REGISTRI per gli operandi

FETCH - DECODE - EXECUTE

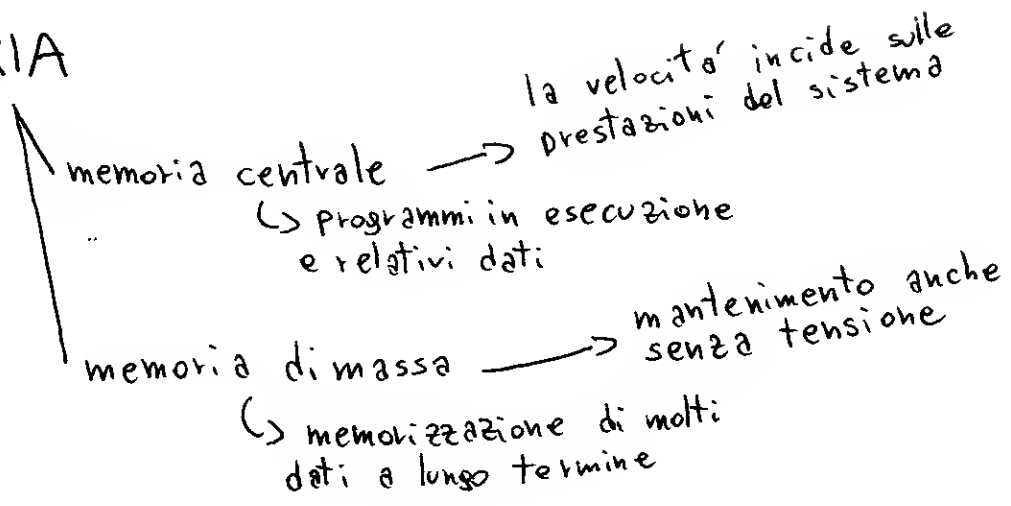
- Segnale di inizio alla CPU, l'unità di controllo fornisce alla memoria l'indirizzo contenente la prima istruzione → scrittura di questo in MAR, attivazione segnale "LEGGI". Trasferimento della memoria a MDR e quindi a IR
- Incremento del contenuto di PC
- DECODE** - Esame dell'istruzione (presente in IR) e determinazione dell'operazione da eseguire
- EXECUTE** - Comando delle unità interessate, prelevando eventuali operandi da memoria, trasferimento risultati nei registri o in memoria
- Ripresa del fetch dell'istruzione successiva

CLASSI DI OPERAZIONI

- trasferimenti CPU ↔ memoria centrale
- trasferimenti CPU ↔ I/O
- elaborazioni dati (op. aritmetiche o logiche)
- controllo del flusso di istruzioni
 - ad esempio, salto da un'istruzione all'altra senza seguire la sequenza

Tutte le operazioni richiedono una
temporizzazione — CLOCK [Hz]

MEMORIA



• CARATTERIZZAZIONE della MEMORIA

- ~ velocità di accesso
- ~ capacità in bit o byte
- ~ volatilità — NON VOLATILE → il dato non viene perso se manca tensione
- ~ costo per bit — VOLATILE → se manca tensione si perde il dato
- rapporto tra costo complessivo e capacità dell'unità

Memorie: elettroniche, magnetiche,
ottiche

ELETTRONICHE → buona velocità, buona capacità,
alto costo e spesso volatilità

MAGNETICHE → basso costo, grande capacità
non volatili, molto lente

OTTICHE → basso costo, grande capacità,
non volatili, scrittura lenta e
complicata (CD-R; CD-RW)

UNITÀ DI MEMORIA

- ~ Tempo di accesso \rightarrow intervallo tra richiesta e fornimento (o scrittura) del dato
- ~ Ciclo di memoria \rightarrow tempo di accesso + intervallo che deve intercorrere fino ad un successivo intervallo di memoria \rightarrow [numero di accessi nell'unità di tempo]
- ~ Velocità di trasferimento \rightarrow larghezza di banda, quantità di dati trasmessi nell'unità di tempo [bit/s; KByte/s]

METODI DI ACCESSO ALLA MEMORIA

- Accesso sequenziale \rightarrow le celle sono posizionate in successione e prima di ciascuna c'è l'indirizzo. La testina di lettura "cerca" il dato leggendo tutti gli indirizzi fino a trovare quello desiderato. Tempo variabile
- Accesso casuale \rightarrow la cella può venir individuata e messa a disposizione in un tempo minimo indipendentemente dalle altre celle \rightarrow RAM
- Accesso misto \rightarrow la cella viene individuata a priori in uno spazio di memoria, entro il quale deve essere effettuata una ricerca (SEQUENZIALE + CASUALE)
- Accesso associativo \rightarrow accesso casuale, nel quale viene confrontato uno o + bit in posizioni specifiche della cella, contemporaneamente a più celle, per raggiungere alte velocità \rightarrow memoria CACHE

MEMORIE A SEMICONDUTTORI

- RAM (Random Access Memory)
- ROM (Read Only Memory) non volatile ma non riscrivibile
- FLASH non volatile e riscrivibile

UTILIZZO NEI CALCOLATORI

- A) Memoria piccola e veloce x CPU
- B) Memoria grande e lenta x dati

RAM

- ↳ SRAM (statiche) → veloci e costose
- ↳ DRAM (Dinamiche) economiche

PRINCIPIO DI LOCALITÀ

Un programma indirizza statisticamente più del 90% delle sue richieste di lettura e ~~scrittura~~ scrittura in un'area di memoria contigua di dimensioni inferiori del 10% dell'area complessiva occupata dal programma e dai dati.

• LOCALITÀ SPAZIALE

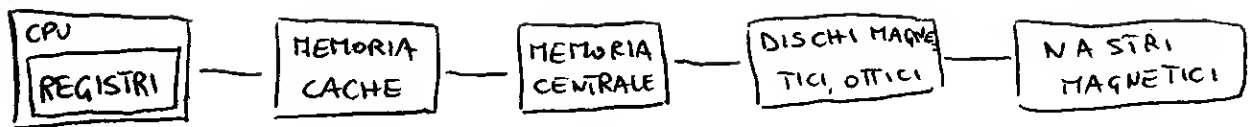
- ↳ Quando un programma fa riferimento a un dato, è probabile che lo stesso programma faccia riferimento in breve tempo ad un altro dato vicino al precedente

• LOCALITÀ TEMPORALE

- ↳ Quando un programma fa riferimento a un dato, è probabile che lo stesso programma faccia nuovamente riferimento a questo dato

UTILIZZO DI UN DATO

- ↳ Spostamento del dato nella memoria veloce



MEMORIE DI MASSA

- grande capacità, non volatilità →
- mantenimento dati per un tempo non definito

- NASTRI E DISCHI MAGNETICI
- DISCHI OTTICI

NASTRI E DISCHI MAGNETICI

- hard disk
 - floppy disk
 - DAT
- supporti ricoperti di materiale magnetico, magnetizzato ~~come~~ opportunamente - formato digitale
- RECORD FISICI separati da INTERRECORD GAP (spazi)
- DENSITÀ → [Bpi]

⇒ Byte per pollice

Cerchi concentrici ⇒ TRACCE

TEMPO DI ACCESSO → Seek time (MEDIO)

floppy → 1,4 Mbyte;
80 tracce e 18 settori

↳ Suddivise in SETTORI

ogni settore ospita un gruppo di RECORD

CREATI ATRAVERSO LA FORMATAZIONE

ACCESSO SEQUENZIALE

DISCHI OTTICI

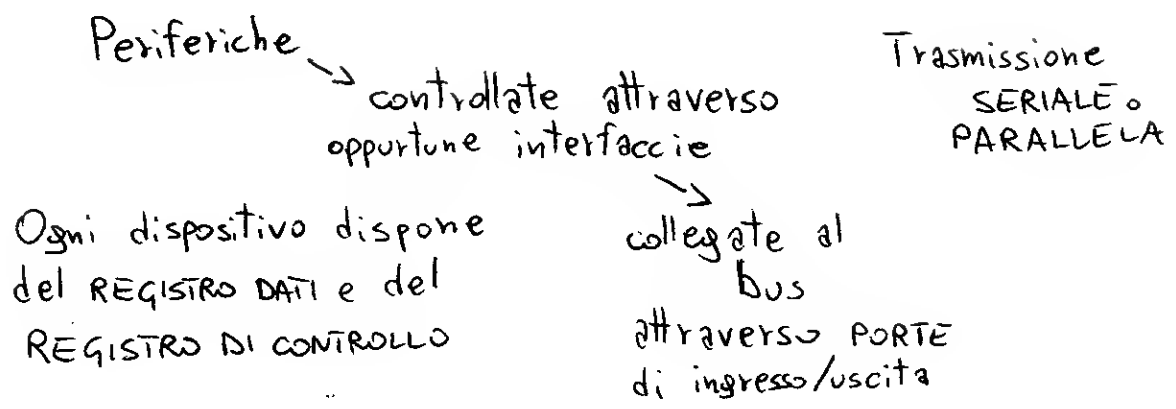
La lettura (e la scrittura) → CD-ROM (CD-R, CD-RW, ...)

DVD (...)

grandi dimensioni

avvengono attraverso un raggio laser che controlla la riflessione (0-1)

INTERFACCIA DI INPUT/OUTPUT



REGISTRO DATI → dati che vengono trasmessi
 " DI CONTROLLO → informazioni sullo stato della periferica
 ↳ SINCRONISMO tra velocità della CPU e della periferica

- 1) CONTROLLO DI PROGRAMMA (POLLING)
- 2) A INTERRUPT
- 3) ACCESSO DIRETTO ALLA MEMORIA

1) POLLING → la CPU esamina iterativamente i registri delle periferiche per verificare se ci sono dati in attesa - LENTEZZA e RISCHIO di PERDITA DATI

2) A INTERRUPT → la periferica, quando ha bisogno di essere servita, invia un segnale alla CPU che interrompe il proprio funzionamento ed esegue il PROGRAMMA DI RISPOSTA ALL'INTERRUZIONE

3) ACCESSO DIRETTO ALLA MEMORIA → si ha un componente hardware specifico (controllore di DMA) solleva la CPU dall'incombenza di inviare i dati, in quanto informato precedentemente sulla locazione di partenza. I dati devono essere contigui.

PERIFERICHE DI I/O

→ strumenti che permettono al calcolatore di acquisire dati e che permettono di presentare i dati generati dalle elaborazioni

VIDEO

dispositivo di output grafico

- LCD (cristalli liquidi)
 - matrice attiva
 - matrice passiva
- CRT (tubo catodico)
 - richiede una potenza di alimentazione maggiore

TASTIERA

principale dispositivo di input → ogni tasto è associato ad una combinazione di segnali elettrici inviati al calcolatore

DISPOSITIVI DI PUNTAMENTO

MOUSE (TRACKBALL) TOUCHPAD

- meccanico
- optomeccanico
- ottico - funzionamento a LED
- Sensori a matrice

STAMPANTI

- AD AGHI - basso costo, - versatilità
- A GETTO D'INCHIOSTRO - basso costo a colori
- LASER - qualità B/N (colori costoso) - velocità

MODEM

Permette trasmissioni di dati su linea telefonica

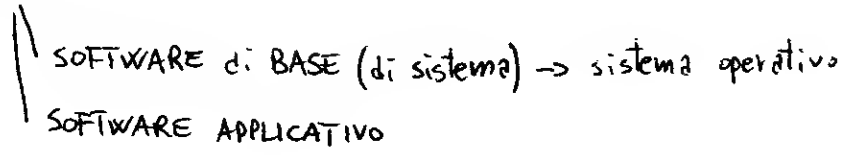
→ Modulazione - DEModulazione dei segnali binari forniti e richiesti dal PC

VELOCITÀ [kbit/s]

- esterno al PC
- interno " "

4- ARCHITETTURA DI UN SISTEMA INFORMATICO. SOFTWARE DI BASE

Software



SISTEMA OPERATIVO

Collezione di programmi interagenti che forniscono all'utente un'astrazione delle operazioni sull'hardware

MODALITA' STANDARD DI INTERFACCIA

- facilitazione dell'utilizzo
- regolamentazione delle risorse

- sviluppo di programmi in modo indipendente allo specifico calcolatore

~~aggiornamento del software~~
two senza

- aggiornamento del software di base senza che gli applicativi e l'utente ne siano influenzati

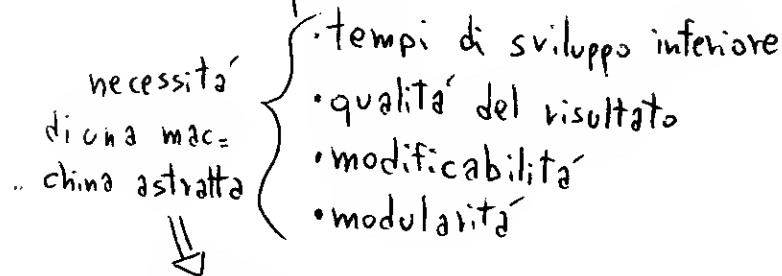
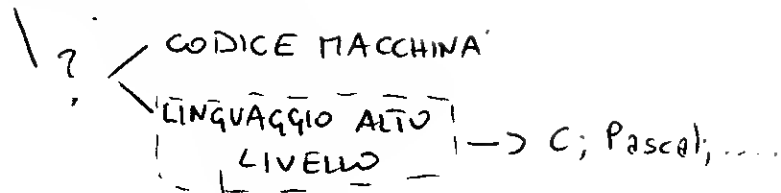
SISTEMA ESTESO, VIRTUALE, PIU' SEMPLICE DA USARE rispetto all'utilizzo diretto dell'hardware

RAPPRESENTAZIONE GRAFICA del sistema operativo

3 cerchi concentrici

- 1°-centrale -> Hardware con il suo codice macchina
- 2° -> Sistema operativo (interprete comandi) -> interfacciamento diretto all' HW
 - SHELL -> ricezione comandi utente
- 3° -> Applicativi -> interfacciati direttamente all'utente

SCRITTURA DI UN PROGRAMMA



INTERPRETE

- 1) Interpretazione linguaggio
- 2) Esecuzione azioni elementari ad essa associate
- 3) Determinazione istruzione successiva

INTERPRETE

- hardware → ad es. la CPU è interprete del proprio linguaggio macchina
- software → traduzione dal linguaggio ad alto livello al codice macchina

interpretazione
compilazione

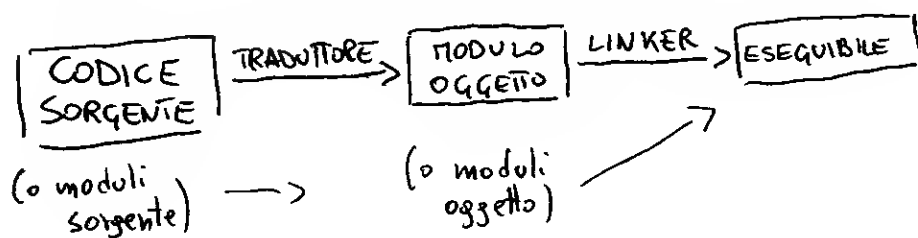
Interpretazione → l'interprete esegue in tempo reale il programma considerando riga per riga, una alla volta

↳ traduzione di tutto il programma ad ogni utilizzo

interprete hardware → + efficiente

" software → flessibile, debug facilitato

Compilazione → La compilazione viene eseguita un'unica volta, e più veloce nell'esecuzione ma ogni modifica richiede la ricompilazione globale



Traduzione → controllo sintassi e redattura delle tabelle di riferimento (procedure, variabili...)

Collegamento (LINKER) → unione di tutti i moduli (se più di uno), creazione tabella di riferimento unica

ESEQUIBILE → scritto nel linguaggio della CPU

EFFICIENZA

interpretazione → programmi che si basano in particolare sull'I/O, perché la velocità di interpretazione è comunque minore del tempo richiesto per l'interfacciamento (PROG. VINCOLATI DA I/O → I/O BOUND)

compilazione → programmi che, rispetto all'I/O, effettuano una grande di elaborazioni (PROG. VINCOLATI DA CPU → CPU BOUND)

ARCHITETTURA DI UN SISTEMA OPERATIVO

Sistema operativo → riconducibile all'architettura di Von Neumann

GERARCHIA A MACCHINE VIRTUALI

↳ Modello a "buccia di cipolla"

- Ogni macchina virtuale è un insieme di programmi che realizza funzionalità sempre più potenti, basate sui livelli inferiori
- Ogni macchina virtuale gestisce determinate risorse, regolamentazione dell'uso secondo modalità di interfaccia ben definite

STRUTTURA DEL S.O., OGNI STRATO È UNA MACCHINA VIRTUALE:

- 1° - hardware
- 2° - nucleo
- 3° - gestore memoria
- 4° - gestore periferiche
- 5° - gestore file (file system)
- 6° - interprete comandi
- 7° - applicativi

NUCLEO

- ! colloquia direttamente con l'hardware
- permette la condivisione delle risorse tra diversi programmi, dando a ciascuno l'utilizzo di una CPU VIRTUALE

→ insieme di unità di elaborazione virtuali dedicate ciascuna ad ogni processo

GESTORE DELLA MEMORIA

destina opportunamente le risorse di memoria → i programmi lavorano in uno spazio virtuale di indirizzamento (come con la CPU)

- ~ protezione dei dati e delle istruzioni, nessun programma può accedere o modificare i dati di un altro processo
- ~ mascheramento della posizione fisica dei dati, in modo che un processo possa anche lavorare con più memoria di quella centrale
- ~ permettere in modo controllato la sovrapposizione dei dati ~~dati~~ condivisi da più processi, in modo da evitare la presenza degli stessi dati in più posizioni diverse → guadagno di risorse

GESTORE DELLE PERIFERICHE

si occupa di tutti gli aspetti che competono l'I/O, fornendo ai processi l'astrazione di lavorare con dispositivi sui quali scrivere o dai quali leggere, senza preoccuparsi dell'aspetto fisico (indirizzamento, sincronizzazione...)

GESTORE DEI FILES

FILE SYSTEM

gestisce le memorie di massa, organizzando i dati in FILES e identificandoli con nome, diritti di accesso, per permettere o meno la condivisione

INTERPRETE DEI COMANDI

↓
| direttamente visibile all'utente
| si occupa di ~~non~~ interpretare i
| comandi giunti dall'utente (tastiera)
| e di attivare i programmi
| corrispondenti.

~ LETTURA della memoria di massa
del programma richiesto → GESTORE DEL FILE

~ ALLOCAZIONE memoria centrale → GESTORE DELLA
MEMORIA

~ CARICAMENTO memoria del programma e
dei relativi dati ↗

~ CREAZIONE e ATTIVAZIONE del processo
corrispondente → NUCLEO

• INTERFACCIA UTENTE GRAFICA (GUI) → facilita
l'utilizzo del sistema ad utenti non esperti.

SUPPORTO DI RETE

| virtualizzazione delle risorse
| ↓
| non si ha più il concetto
| di localizzazione delle risorse

visione unificata dei dati,
anche da postazioni a distanza

PARALLELISMO

Pur rappresentando il sistema di Von Neumann un'esecuzione sequenziale delle operazioni, può essere vantaggioso considerare la possibilità di parallelismo tra le azioni svolte

ES.

~livello di DATI → negli algoritmi per il trattamento di immagini, i pixel possono venir trattati in modo indipendente e quindi contemporaneamente

~livello di ISTRUZIONI → istruzioni che svolgono operazioni tra loro indipendenti possono essere svolte contemporaneamente

~livello di PROGRAMMI → più programmi possono venir eseguiti nello stesso istante

Ai fini di un parallelismo efficace, il sistema operativo deve amministrare un insieme di risorse scarse rispettando le condizioni:

- EFFICIENZA → è richiesto un utilizzo ottimale di tutte le risorse
- INTERATTIVITÀ → il tempo di risposta deve rientrare in tempi accettabili
- COOPERAZIONE → il sistema deve essere gestito da più agenti, onde evitare che il malfunzionamento di un programma porti al blocco completo del sistema

PRESENZA DI UNA UNICA CPU \rightarrow SIMULAZIONE DEL
PARALLELISMO
Processi

PROCESSI \equiv PROGRAMMA

Ad un programma corrisponde uno o più processi

(ad esempio:

A) videoscrittura

A2) stampa documento)

Distinzione dello stato
dei processi.

\rightarrow ESECUZIONE \rightarrow ha la CPU a disposizione
per l'effettuazione di calcoli
(max 1 nei sistemi uniprocessore)

\rightarrow PRONTO \rightarrow in grado e pronto ad essere
eseguito, ma in attesa di \rightarrow coda
essere messo in esecuzione FIFO

\rightarrow IN ATTESA \rightarrow per passare nello stato di
"pronto" necessita di un evento
esterno

Per evitare che un processo resti in esecuzione ritardando
il resto, dopo un tempo prestabilito ~~viene salvato~~ viene
salvato il suo "contesto" e messo nella coda FIFO come ultimo,
per essere ripreso in considerazione quando la coda si
estingue. Un processo viene distrutto quando genera il
comando di termine.

MULTIPROGRAMMAZIONE

Sistemi operativi che permettono l'esecuzione di un unico applicativo alla volta → UNIPROGRAMMATI
(MS-DOS)

Sistemi operativi che permettono l'esecuzione di più programmi per volta → MULTIPROGRAMMATI

modalità ROUND-ROBIN
(time-sharing)

Vengono sfruttati i tempi morti (come l'attesa di input) per eseguire a rotazione gli altri processi in stato di pronto.

POSSIBILITÀ di MIGLIORAMENTO

- calcolatori con più di una CPU
- distribuzione lavoro su CPU di calcolatori connessi in rete
- accesso contemporaneo ad un'unica risorsa da parte di più utenti
- condividere la stessa risorsa fisica

INCONVENIENTI

PROCESSI:

- in foreground → attivo e abilitato all'interazione con l'utente

- in background → attivo ma temporaneamente non in grado di interagire con l'utente

• starvation → impossibilità di accesso a una risorsa a causa del kernel (come assegnamento di priorità ai processi)

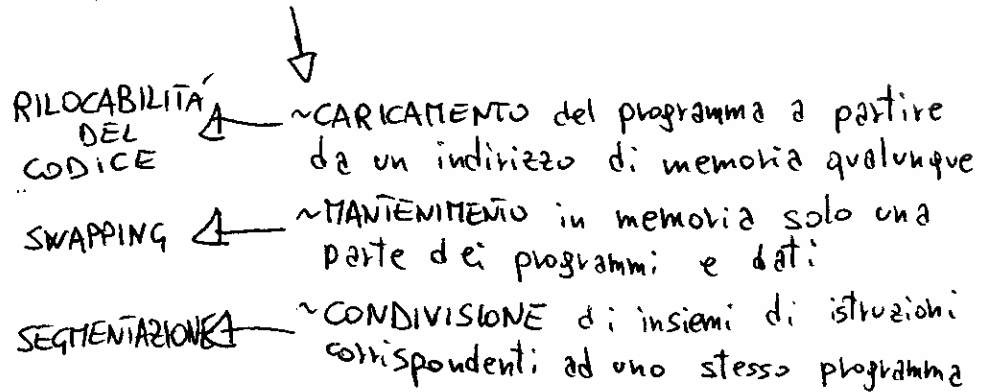
• blocco critico → più processi si bloccano
(DEADLOCK) ↳ vincolo circolare

Soluzione

↳ eliminazione dei processi coinvolti
↳ verifica preventiva

GESTIONE DELLA MEMORIA CENTRALE

Nel supporto della multiprogrammazione, lo spazio richiesto dai vari processi può diventare maggiore dello spazio effettivamente disponibile nella memoria centrale

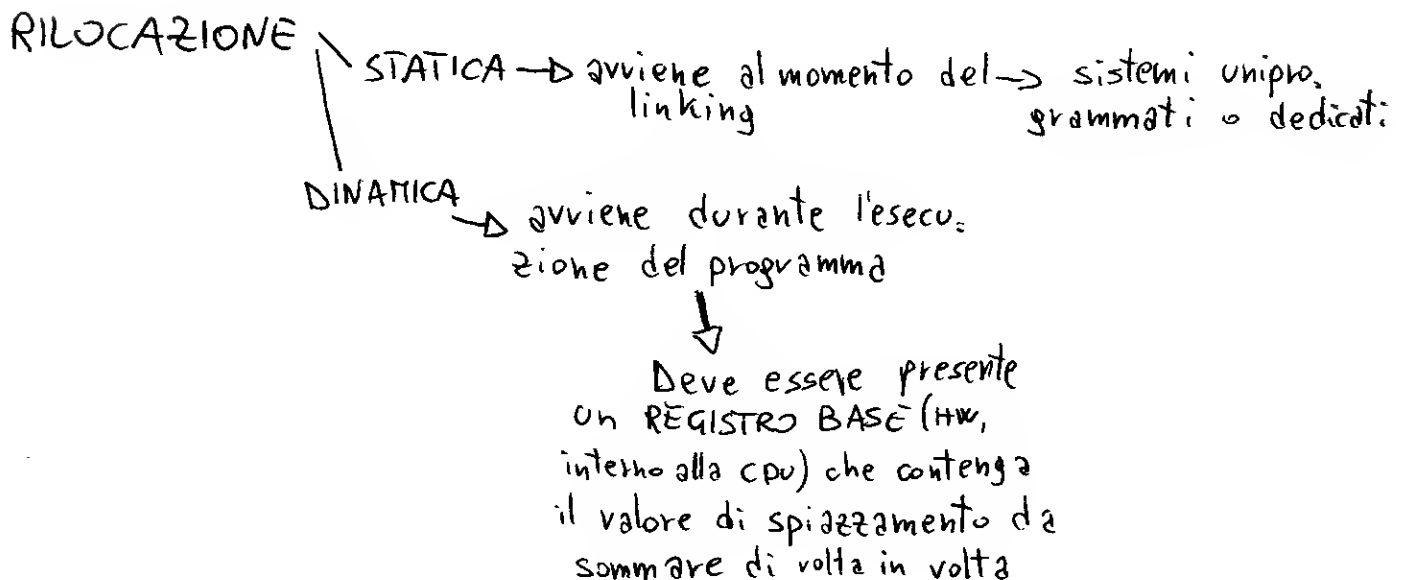


RILOCABILITÀ DEL CODICE

Nella fase di compilazione e di linking, tutti i riferimenti sono stati risolti, contengono cioè indirizzi di memoria, presupponendo un intervallo di memoria (spazio logico) avente come inizio la cella 0.

↓

Siccome nell'esecuzione quest'idealità non viene mai rispettata, si deve procedere allo SPIAZZAMENTO (RILOCAZIONE)



SWAPPING

Nel momento in cui la memoria centrale non dispone di sufficiente spazio per i processi attivi, onde evitare il termine forzato di questi processi vengono spostati (quelli in attesa e quelli pronti) in un'area della memoria di massa, detta "DI SWAP".

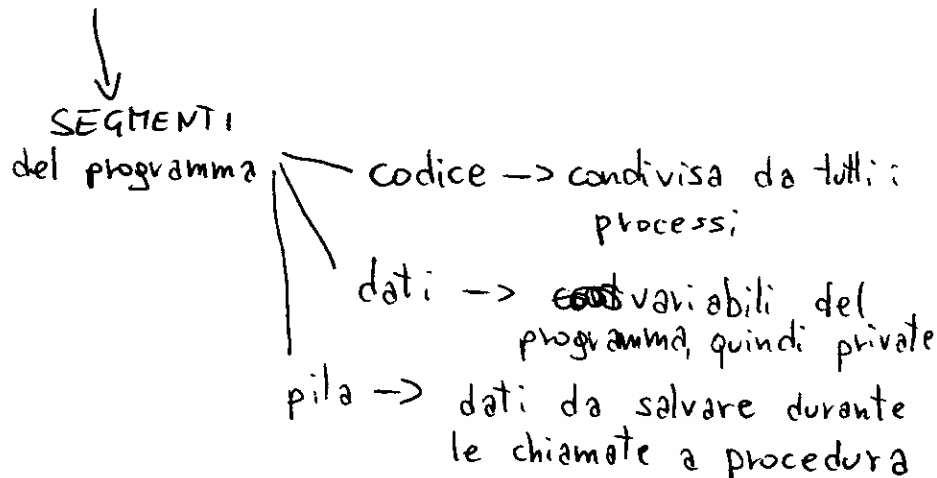
↳ Suddividendo il programma in sezioni di dimensioni fisse (PAGINE LOGICHE) ed organizzando di conseguenza la memoria (PAGINE FISICHE) si ha un miglioramento dello swapping

MODO
PAGINATO ←

- Estensione delle dimensioni di un processo utilizzando zone di memoria anche non contigue
- Mantenimento in memoria solo della porzione di ~~programma~~ desiderato

SEGMENTAZIONE DELLA MEMORIA

Il programma viene suddiviso distinguendo i dati (riservati) alle istruzioni che, essendo le stesse, possono essere raccolte una volta sole ed utilizzate da chiunque lo richieda



GESTIONE DELLA MEMORIA DI MASSA

FILE SYSTEM

gestione della
memoria di massa

- Recupero di informazioni precedentemente memorizzate
- Eliminazione delle informazioni obsolete
- Modifica delle informazioni esistenti
- Copia delle informazioni

LOCALIZZAZIONE DEI CONTENUTI

- filename + estensione → nomefile.est
- ordinamento in directory (cartelle)
con una struttura ad albero → PERCORSO

Identifica il tipo
di file

- ASSOLUTO → Dalla radice
- RELATIVO → Da una qualsiasi cartella

SERVIZI DI BASE

↓
Modalità differenti per
utente e programmi

- VISUALIZZARE informazioni sul contenuto
- OPERARE (creare, cambiare nome, cancellare...)
- VISUALIZZARE secondo attributi definiti
(es. data di creazione)

COLLEGAMENTI (shortcut)

↙ a file
↘ a cartelle

→ Rende visibile una
risorsa in un punto dove
essa in realtà non
è presente

↓
RISPARMIO SPAZIO
ORGANIZZAZIONE

FILE SYSTEM PER MULTIUTENZA

(22)

Insieme di calcolatori connessi in rete

↓
Richiesta ASTRAZIONE dal MODELLO
fisico dell'organizzazione files

- ↓
- ~ integrazione files di una stessa rete
 - ~ soluzione all'univocità dei nomi dei files e delle cartelle
 - ~ accesso alle risorse presenti su calcolatori remoti

ORGANIZZAZIONE FISICA DEI DATI

MS-DOS, Windows...

↓
Struttura a
LISTA
CONCATENATA → memorizzata nella tabella
di allocazione dei file

FAT

→ contiene anche un
descrittore di files
(data, ora...)

GESTIONE DELL' I/O

- ! driver fisici → HW, effettuano a livello fisico i trasferimenti
- driver logici → SW, maschera ai livelli superiori le complesse operazioni astruendo le operazioni

PLUG AND PLAY
PnP

⇒ La periferica collegata al calcolatore viene automaticamente riconosciuta e configurata, senza l'intervento dell'utente nella configurazione

SICUREZZA E PROTEZIONE

Tipicamente, per l'accesso ad un sistema operativo (in particolar modo se è connesso in rete) è necessario un account, identificante l'utilizzatore e protetto da una password

Accesso "root"
(AMMINISTRATORE)

↓
potere su
tutte le
risorse

↘
GESTIONE DEL
SISTEMA

PERSONALIZZAZIONE



- distribuzione dei costi di gestione in base alle risorse utilizzate dall'utente
- porzione di file system visibile e di periferiche disponibili
- personalizzazione dell'ambiente

TASSONOMIA DEI SISTEMI OPERATIVI

CLASSIFICAZIONE UTENTI

~ Programmatori di sistema → Realizzano applicazioni, di conseguenza hanno una visione completa del sistema ed hanno accesso alle librerie di funzionamento del sistema operativo

~ Amministratori di sistema → Gestiscono il corretto funzionamento del sistema, creano account per gli utenti e selezionando i relativi diritti, eseguono copie di backup

~ Utenti applicativi → Utilizzano il sistema ed i programmi messi a disposizione, non hanno conoscenza specifica, con la semplicità delle operazioni la produttività cresce

SISTEMI A LOTTI (BATCH)

Sono sistemi in cui è ottimizzata l'elaborazione grazie ad una precisa distribuzione temporale. Esso deve svolgere dei "lavori" (job), raggruppati in lotti. I lotti vengono disposti in coda ed eseguiti. L'I/O interessa solo la memoria di massa, in quanto viene eliminato il rapporto con l'utente. La CPU non è disponibile sino al termine dei calcoli; il tempo di latenza è notevole. → CALCOLI SCIENTIFICI

SISTEMI DEDICATI

Tutte le risorse sono dedicate ad un unico utente o comunque ad un unico scopo. Il sistema operativo è semplice ed ha scarsa portabilità verso altre piattaforme; si ha un basso sfruttamento perché rimane spesso inattivo → CENTRALINE AUTOMOBILI
→ BANCOMAT

SISTEMI INTERATTIVI

Sono sistemi caratterizzati dalla presenza di più utenti, ognuno dei quali ha a disposizione un proprio terminale (video e tastiera) che competono per l'utilizzo delle risorse. È importante il TEMPO DI RISPOSTA → attuazione di tecniche di TIME-SHARING. Il sistema operativo è complesso; la CPU è sfruttata al massimo ed ogni utente opera come se avesse un calcolatore dedicato

SISTEMI IN TEMPO REALE

Sistemi fortemente interagenti con l'esterno che sono caratterizzati dal tempo di risposta, alla quale è legata la funzionalità. La richiesta di un certo tempo dipende dall'utilizzo al quale il sistema è progettato. → CONTROLLO INDUSTRIALE

SISTEMI TRANSAZIONALI

|
Caratterizzati dall'interattività, legati principalmente all'interrogazione e all'aggiornamento di archivi → TEMPI DI RISPOSTA BREVI

~ Criticità → necessità di evitare malfunzionamenti

~ Numero potenziali clienti e dislocazione geografica

ATOMICITÀ → due modi per terminare la transazione

- |
- COMMIT → corretto, modifica persistente dati
 - ABORT → errore, ripristino del dato di partenza

PERSISTENZA → ogni transazione a buon fine deve produrre effetti permanenti

ISOLAMENTO → indipendenza di una transazione rispetto alle altre



SISTEMI
BANCARI

MODELLI ORGANIZZATIVI DEI SISTEMI OPERATIVI

processi ——— temporanei
 permanenti

- Monolitico
- A strati
- Client-server
- Modelli ibridi

MODELLO MONOLITICO

Un solo processo che provvede all'esecuzione di una serie di procedure per la gestione del sistema

MODALITÀ DI FUNZIONAMENTO

utente → normale esecuzione, non si può accedere a tutte le risorse
supervisore (kernel) → svolgimento di particolari operazioni, non si hanno limitazioni

⇓
 sistemi semplici,
 es. controllo di
 una linea di produzione

MODELLO A STRATI

in gerarchia a strati, ognuno dei quali costruito sul sottostante

⇒ Separazione meccanismi e politiche di gestione

1. processi operatore
2. " livello utente
3. gestione I/O
4. comunicazione utente/processi
5. gestione memoria (anche swap)
6. gestione processi (CPU)

⇓
 ALLOCAZIONE IN
 AREE COMUNI

⇓
 PORTABILITÀ
 (tranne nucleo)

Assenza di protezione dati relativi alle risorse

ARCHITETTURE CLIENT-SERVER

nucleo = server

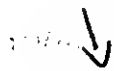
processi = client

⇒

Il client conosce le modalità di accesso alle risorse, comunica tuttavia con il server che esegue quanto richiesto e spedisce al client il risultato ⇒ SCAMBIO DI MESSAGGI

- indipendenza fisica tra server e client → protezione server
- possibilità di specializzare un componente in maniera trasparente all'altro
- minima manutenzione software → le parti comuni sono nel server

MODELLI IBRIDI



tra modello a strati e client-server



Suddivisione processi in gruppi, i processi nello stesso gruppo sono condivisi; per gli altri si ha uno scambio di messaggi



SEMPLIFICAZIONE
INTERAZIONI TRA PROCESSI,
AUMENTO DELLO
SFRUTTAMENTO DELLA CPU

WINDOWS NT

Windows NT Workstation
" " Server

> il sistema può operare nei modi

• kernel (privilegiata)
• utente

Eventuali malfunzionamenti di un programma non compromettono la stabilità del server

- funzionalità di comunicazione e interfaccia tra le varie applicazioni
- tutta la memoria è accessibile
- tutte le istruzioni della CPU

→ ISOLAMENTO FRA
APPLICAZIONI

Windows NT Executive

→ componenti S.O. che si interfacciano all'hardware

MODULARITÀ, INCAPSULAMENTO, PROTEZIONE



La dimensione della memoria virtuale è dinamica e stabilita da HAL (Hardware Abstraction Layer)

Solo una parte del sistema è delegata ad una specifica funzione, impedendo a qualsiasi altro software l'accesso

5- RETI DI CALCOLATORI

ELABORAZIONE e DISTRIBUZIONE DELLE INFORMAZIONI

Mainframe - che condivideva le risorse in tanti terminali: → i programmi erano residenti ed eseguiti dal mainframe stesso

- RETI DI CALCOLATORI → ogni postazione è indipendente e, in caso di interruzione della rete, può lavorare con gli strumenti di cui dispone
- SISTEMI DISTRIBUITI →
 - gli utenti non hanno visibilità sull'architettura, il sistema è omogeneo e progettato per un'unica applicazione → BANCOMAT
 - ↳ interruzione della rete → inutilizzabilità

FINALITÀ DI UNA RETE

- condivisione risorse (dati, programmi, hardware) → RISPARMIO
- comunicazione tra utenti (posta elettronica)
- miglioramento dell'affidabilità (~~disponibilità~~ di risorse alternative)
- risparmio (decentramento e condivisione risorse)

gestione risorsa → server di rete



- gestione files
- " stampe
- " comunicazioni

NECESSITÀ DI CONDIVISIONE A PARTIRE DAL BASSO ⇒ evoluzione bottom-up

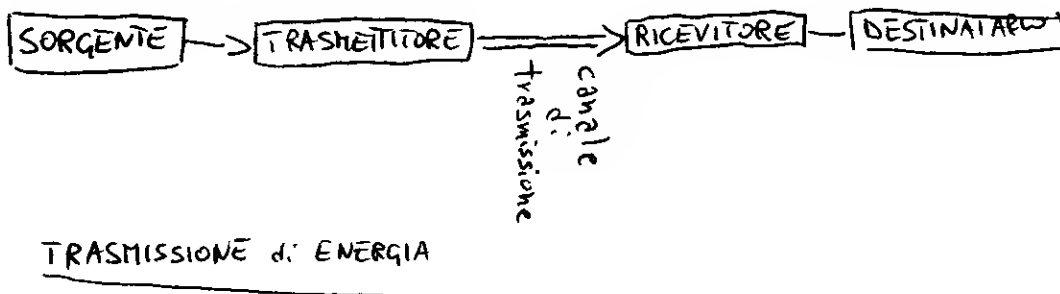
TIPOLOGIE DI COMUNICAZIONE

- multipunto (BROADCAST) → canale condiviso da tutti i calcolatori, ognuno identificato da un INDIRIZZO DI RETE
 - punto-a-punto → ~~serie~~ connessioni individuali tra ogni coppia di calcolatori
- difficoltà se è previsto un collegamento dei calcolatori non continuativo
- ↓
- complessità
 - costo elevato

DISTINZIONE IN BASE ALLE DIMENSIONI

- LAN (Local Area Network) → collegamenti nello stesso edificio
- MAN (Metropolitan Area Network) → collegamenti nella stessa area urbana
- WAN (wide Area Network) → collegamenti in un'area geografica
- Internetwork → unione di più reti geografiche
⇒ INTERNET

COMUNICAZIONE DEI DATI



CASI DI TRASMISSIONE

~ MESSAGGI ANALOGICI SU CANALE ANALOGICO — il segnale occupa lo stesso spettro
MODULAZIONE

~ MESSAGGI ANALOGICI SU CANALE DIGITALE — DIGITALIZZAZIONE

~ MESSAGGI DIGITALI SU CANALE ANALOGICO — trasformazione DIGITALE → ANALOGICO

~ MESSAGGI DIGITALI SU CANALE DIGITALE — operazioni di corrispondenza tra bit e segnale inviato

Nella trasmissione si hanno interferenze che creano disturbi nelle informazioni ⇒ BIT di CONTROLLO per ricreare il dato originale

MEZZI DI TRASMISSIONE

• mezzi guidati → linee fisiche — doppino telefonico
cavo coassiale
fibra ottica
• mezzi non guidati → irradiazione — antenna

CARATTERISTICHE DEL MEZZO

- capacità di canale → velocità di trasferimento [bit/s]
- attenuazione del segnale → se troppo elevato, uso di ripetitori posti lungo la linea
- interferenze tra segnali → assenza o presenza di schermatura
- numero di ricevitori → tasso di distorsione se il mezzo è condiviso da più comunicazioni

MEZZI DI TRASMISSIONE GUIDATA

DOPPIO TELEFONICO

Poco costoso, largamente utilizzato
Segnali sia analogici che digitali
Costituito da due fili di rame sin-
golarmente isolati \Rightarrow max 4-10 Mbit/s

CAVO COASSIALE

Corpo centrale conduttore, protetto
da un corpo isolante, una maglia
per la schermatura ed una guaina
isolante esterna \Rightarrow max 500 Mbit/s

FIBRA OTTICA

Sottile materiale entro il quale un
raggio luminoso, prodotto dalla sorgente,
viene rifratto all'interno del
mezzo \Rightarrow max 2 Gbit/s, 2 GHz

MEZZI DI TRASMISSIONE NON GUIDATA

Antenne \rightarrow irradiazione e captazione

FREQUENZE

- 30 Mhz : 1 GHz \rightarrow non direzionali
- 2 GHz : 40 GHz \rightarrow microonde, direzionali (via satellite)
- 300 GHz : 200 THz \rightarrow infrarossi, multipunto in aree limitate

TECNOLOGIA DI TRASMISSIONE



- Sincrona → sincronizzazione tra trasmettitore e ricevitore, impostazione a priori delle caratteristiche
- Asincrona → presenza di BIT di START e BIT di STOP, la trasmissione avviene un carattere per volta
- ~ FULL DUPLEX → trasmissione contemporanea in entrambe le direzioni
- ~ HALF DUPLEX → trasmissione alternata tra una sorgente e un ricevitore che si alternano

MULTIPLEXING

✓ condivisione di un unico mezzo per più canali

• TDM (Time Division Multiplexing) → il mezzo viene messo alternatamente a disposizione

✓ • FDM (Frequency Division Multiplexing) → attribuzione di un differente spettro di frequenze

STDM (Statistical TDM) → suddivisione dei tempi su base della disponibilità dei dati da trasmettere

RETI GEOGRAFICHE

WAN

Uniscono reti di calcolatori distanti tra loro

- 1) Rete di calcolatori
- 2) Rete di trasmissione

IMP (Interface Message Processor)

Si occupano dell'instradamento dei dati

Instradamento
Host → IMP → IMP → Host

Rele
COMMUTATA

collegati a Host che comunicano direttamente ai calcolatori in rete

COMMUTAZIONE DI CIRCUITO

Da parte della sorgente viene stabilito un percorso per raggiungere il destinatario. Il percorso viene quindi occupato fino al termine della comunicazione

⇒ COMUNICAZIONI TELEFONICHE

↙ inutilizzabilità da parte di altri

↘ richiesta velocità tra calcolatori corrispondente

COMMUTAZIONE DI PACCHETTO

I dati vengono inviati suddivisi a pacchetto, ognuno dei quali contiene l'indirizzo del destinatario.

Ogni nodo memorizza il pacchetto e lo invia nel percorso libero - l'ultimo nodo ricompone il dato

- ⇒
- Utilizzo efficiente delle linee
 - Anche in caso di traffico elevato non si ha congestione
 - Possibilità di gestione priorità diverse
 - Indipendenza dalla velocità di ogni calcolatore

FRAME RELAY

Si utilizza una tecnologia a commutazione di pacchetto, eliminando i dati di controllo resi inutili dal miglioramento della qualità delle trasmissioni

⇒ in caso di errori, il destinatario chiede la ripetizione dell'invio

ISDN

Utilizzo sia della commutazione di pacchetto che della commutazione a circuito e di una segnalazione di canale comune che permette di controllare connessioni multiple utilizzando cammini diversi



64 Kbit/s

• ACCESSO BASE \rightarrow 2 canali a 64 Kbit/s e un canale di servizio a 16 Kbit/s

• " PRIMARIO \rightarrow

30 canali a 64 Kbit/s

1 canale di servizio a 16 Kbit/s

ATM

(Asynchronous Transfer Mode)

COMUTAZIONE di CIRCUITO e di PACCHETTO

- dati digitali
- simile al frame relay \rightarrow organizzato in CELLE di lunghezza fissa
- sia connessioni continue (videofonia)
" discrete (dati)

125 - 155 Mbit/s

RETI LOCALI

Connessione Broadcast \Rightarrow ogni postazione ha un'unità di trasmissione/ricezione collegata ad un canale condiviso

~ AMPIA LARGHEZZA DI BANDA

~ MODULARITÀ e FACILITÀ \Rightarrow e' suff. una scheda di interfaccia

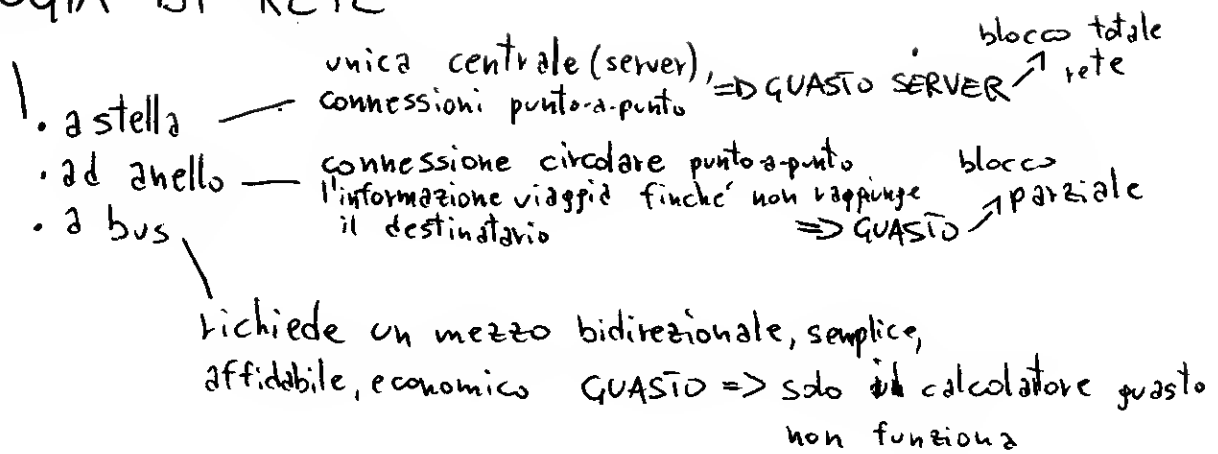
~ AFFIDABILITÀ

~ ESPANDIBILITÀ

~ ECONOMICITÀ

- Dati in formato digitale attraverso codifica
 - \rightarrow ASCII per i caratteri
 - \rightarrow PCM (Pulse Code Modulation) per i segnali analogici

TOPOLOGIA DI RETE



METODI DI ACCESSO

Nella condivisione del mezzo di trasmissione è necessario che non avvengano collisioni (e quindi perdite) di dati:

- TECNICHE A CONTESA → Risolvere le collisioni
- EVITARE in partenza LE COLLISIONI

TECNICHE A CONTESA (CSMA-CD)

↓ Carrier Sense Multiple Access with Collision Detection → ADOTTATO DA ETHERNET

prevede i calcolatori posti in parallelo

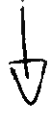
→ Non appena il canale è libero, i dispositivi che hanno dati in attesa trasmettono → Se si verifica una collisione, viene sospeso l'invio che si SFASAMENTO DEI DATI
→ ripete dopo un tempo casuale

Tempi di propagazione diversi da

TECNICHE NON A CONTESA (token ring e token bus)

Token ring → calcolatori ad anello

Token bus → calcolatori a bus



I calcolatori assumono a turno il diritto di scrivere sulla linea, scambiandosi un dato codice

✓
garanzia di invio entro un tempo limite ⇒ Utilizzato per applicazioni temporaneamente critiche

TIPOLOGIE DI RETE

-CLIENT-SERVER → È presente una postazione dedicata (SERVER) che gestisce la condivisione tra le risorse

Dai client è possibile utilizzare programmi e dati presenti nel server ed i dati per i quali è permesso l'accesso. Normalmente non è possibile accedere da un client alle risorse di un altro client ⇒ CONTROLLO CENTRALIZZATO

-PEER-TO-PEER → Tutti i calcolatori sono collegati tra loro, ognuno lavora con le risorse presenti sul proprio e cura la condivisione con gli altri ⇒ CONTROLLO DELEGATO A CIASCUN UTENTE

INTERCONNESSIONE DI LAN

È possibile connettere più LAN dello stesso tipo attraverso un BRIDGE, che effettua lo scambio di dati come se le LAN fossero una unica

COMPONENTI DI UNA RETE

- componenti software → driver e programmi di scambio dati, software presenti da condividere
- " hardware → integrazione dei dispositivi hardware
- " di interconnessione → cablaggio, da effettuare secondo la configurazione scelta

ARCHITETTURA DI COMUNICAZIONE

Per poter stabilire una connessione funzionale è indispensabile definire le caratteristiche di comunicazione

PROTOCOLLI DI COMUNICAZIONE

stabiliscono:

- formato dei dati
- struttura dei pacchetti
- velocità

Definizione di

• Ogni protocollo assume un ← un insieme di protocolli; maggiore flessibilità

MODELLO ISO-OSI

MODELLO DI RIFERIMENTO
DELL'ARCHITETTURA DI UN SISTEMA
DI COMUNICAZIONI DA CALCOLATORI

↳ Suddivisione in
7 livelli di ogni stazione
← in rete

Descrizione delle
caratteristiche delle
architetture di reti

APPLICAZIONE
PRESENTAZIONE
SESSIONE
TRASPORTO
RETE
DATA LINK
FISICO

Livelli più alti → generazione di un messaggio



Data link → suddivisione in blocchi di dimensioni minori (FRAME)



Fisico → trasferimento bit a bit

LIVELLO FISICO

→ trasmissione dei singoli bit

definisce:

- quale canale utilizzare considerando la velocità richiesta e la topologia della rete di comunicazione
- forma di trasmissione (ANALOGICA/DIGITALE) →
→ indipendente dalla natura originaria del file
- direzione del flusso di dati
 - └ SIMPLEX
 - └ HALF DUPLEX
 - └ FULL DUPLEX
- interfaccia da utilizzare per il collegamento

LIVELLO DATA LINK

→ comunicazione tra nodi adiacenti: controllo della collisione dei dati

- gestione del canale di collegamento (MAC → Medium Access Control)
- strutturazione in frame (LLC → Logical Link Control)



generazioni di "frame" in base
al tipo di connessione utilizzato

(token bus, token ring, ethernet)



identificare &
ripetere l'utilizzo
in caso di
errore

LIVELLO RETE

- Si occupa della gestione della trasmissione

~ Reti broadcast

Ogni calcolatore riceve il messaggio, esamina l'indirizzo di destinazione e decide se eliminarlo o no

~ Reti punto-a-punto

È necessario individuare un percorso lungo il quale instradare il messaggio

PERCORSO: Stabilito in maniera

↳ statica ⇒ Parametri fissati a priori

• dinamica ⇒ Analisi situazione rete, minor congestione



maggior complessità algoritmo

LIVELLO TRASPORTO

- Stabilisce SESSIONI DI COMUNICAZIONE tra utenti collegati e calcolatori nella rete

Sistemi evoluti utilizzabili dai livelli più alti (ad es. trasferimento files)

⇒ DIALOGO TRA DUE CALCOLATORI

- Stabilisce a chi spetta il canale
- Sincronia sorgente-destinatario
- Gestire chiamate di procedure remote (RPC) ⇒ client/server

LIVELLO PRESENTAZIONE

(31)

Svolge una serie di funzioni richieste frequentemente da una rete di calcolatori tanto da preferire una realizzazione unica piuttosto che distribuita fra + programmi:

- CODIFICA dati standard
- COMPRESSIONE dati
- CRITTOGRAFIA

LIVELLO APPLICAZIONE

Programmi che sfruttano i servizi offerti dal livello presentazione

- ~ trasferimento file
- ~ terminali virtuali

FILE SERVER VIRTUALE

↳ l'interfaccia dei vari file server

viene ~~resa~~ resa omogenea → standardizzazione delle procedure di accesso

Servizio e-mail

- 1) scrittura e lettura messaggi
- 2) invio e ricezione messaggi (invisibile)

PROTOCOLLI TCP/IP

Transmission Control Protocol/
Internet Protocol

INTEROPERABILITÀ TRA RETI FISICHE DIVERSE

• Protocollo standard per Internet

• Architettura a 5 livelli → FISICO
ACCESSO ALLA RETE ⇒ data link ⇒ gestione instradamento
INTERNET ⇒ formato pacchetti, gestione percorsi
TRASPORTO ⇒ trasferimento affidabile
APPLICAZIONE ⇒ 6/7 osi ⇒ comunicazione con applicazioni

- Successi -

- applicazioni client-server affidabili
- condivisione informazioni tra organizzazioni diverse connesse tra loro
- implementato in molti s.o.

TCP/IP ⇒ Nasconde i dettagli delle reti fisiche,
definisce

- formato indirizzi
- " pacchetti
- tecniche di trasferimento

Indirizzamento univoco



Indirizzo IP → pacchetto di 32 bit

- rete fisica
- singola stazione nell'ambito della rete

IP → Controllo correttezza

✓ INTESTAZIONE

TCP → controllo integrità ✓

DATI → bit aggiuntivi
[checksum]

Connessione reti TCP/IP

• Gateway

• Router

> indirizzano i pacchetti

Gateway ↘

Software per
instradamento IP, e'
dotato di 2 o più
adattatori di rete

• nella rete locale

• ad altre reti ↘

controllo instrada-
mento (se statico)

INCREMENTO CALCOLATORI
CONNESSI AD INTERNET



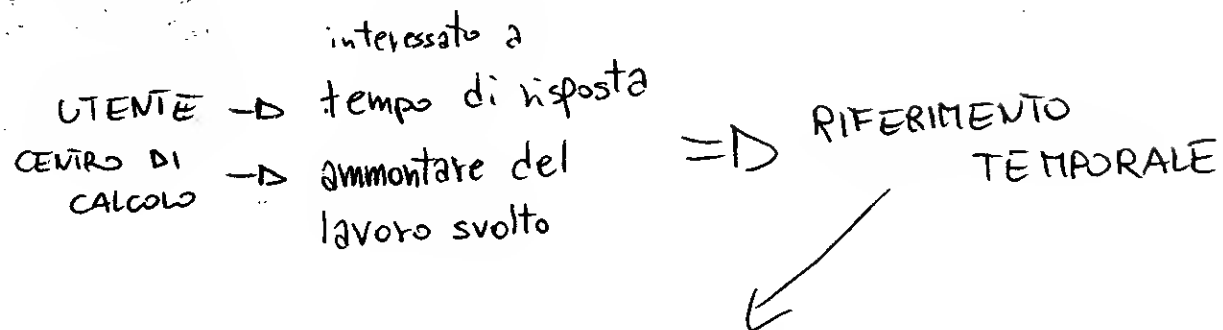
IP a 32 non più
sufficiente



IPv6 → 128 bit
sup. A/V

CRITERI DI CARATTERIZZAZIONE DEI CALCOLATORI

VALUTAZIONE DELLE PRESTAZIONI



TEMPO DI CPU

- 1) frequenza
- 2) n° di clock \times un'istruzione
- 3) n° di istruzioni \times un processore

• elapsed time \rightarrow tempo \times l'esecuzione di un determinato compito

• CPU time \rightarrow tempo \times l'esecuzione riferita alla CPU

FONDAMENTALE LA STRUTTURA

- Pentium
- PowerPC
- Pentium II

~MIPS

Mega Instructions Per Second \rightarrow 1) dipende dalle istruzioni

\rightarrow dipende dal programma

non sempre veritiero: se c'è un coprocessore matematico, le operazioni sono minori, più complesse e si ha maggiore velocità di calcolo

BENCHMARK

TASSONOMIA SISTEMI INFORMATICI

- | | |
|--------------------|----------------------------------|
| • PC | → unico utente, sw standard |
| • Work station | → pochi utenti, " " |
| • Minicalcolatore | → 10-50 utenti, " " |
| • Mainframe | → elevato, sw sviluppato apposta |
| • Supercalcolatore | → " " " " |

~ PC

- Uso personale
- Architettura modulare
- Spesso hanno 1 processore
- Microsoft

~ Workstation

- Utilizzo professionale
(app. scientifiche, ingegneristiche)
- Uno o pochi utenti

~ Minicalcolatore

- Più utenti
- Condivisione dati e sw comuni
(banca, ...)

~ Mainframe

- Elevata capacità di calcolo
- Elevato n° di periferiche
- Batch => coinvolgimento grandi basi dati
- Non richiedono interazione con l'utente frequente

~ Supercalculator

- Struttura interna specifica per complesse elaborazioni
- Grandi centri di ricerca

~ Sistemi multiprocessori:

- Reale parallelismo delle operazioni
↳ più di un processore

- Catena di PIPELINE

↓

Coda di elaborazioni da
completare inviate alla prima
CPU disponibile

CLASSI DI PARALLELISMO

ASSI DI PARALLELISMO

• SISD	(flusso	singolo	di	istruzioni,	flusso	singolo	di	dati)	→ Von Neumann
• SIMD	("	"	"	"	multiple	"	") ⇒ Calcolatori paralleli
• MISD	("	multiple	"	"	singolo	"	"	
• MIMD	("	"	"	"	multiple	"	"	